Ontological Concepts of Non-Functional Requirements Applied to Particular Situations in Measurement and Evaluation Projects

Luis Olsina
Facultad de Ingeniería, UNLPam, GIDIS_Web
General Pico, Argentina, L6360
olsinal@ing.unlpam.edu.ar

and

Pablo Becker
Facultad de Ingeniería, UNLPam, GIDIS_Web
General Pico, Argentina, L6360
beckerp@ing.unlpam.edu.ar

and

María Fernanda Papa
Facultad de Ingeniería, UNLPam, GIDIS_Web
General Pico, Argentina, L6360
pmfer@ing.unlpam.edu.ar

and

Gustavo Rossi
LIFIA, Facultad de Informática, UNLP / CAETI, Facultad de Tecnología Informática, UAI
La Plata, Argentina, B1900
gusrossi@gmail.com

Abstract

The terms Functional Requirement and Non-Functional Requirement (NFR) are intertwined concepts but aim at different purposes. In a nutshell, functional requirements state what an entity must do by referring to its features and capabilities (functions), while non-functional requirements state quality aspects that this entity must achieve. The present paper discusses the main terms and relationships for the Non-Functional Requirements (NFRs) domain ontology, which is located at the top-domain level in the context of a five-tier ontological architecture for specifying knowledge. In this ontological context, the term NFR does not represent a concrete entity (particular thing), but rather a quality-, constraint-related human assertion of a certain aspect of the entity to be measured and evaluated. Also, an NFR as an assertion can deal with an entity category (thing category) as a universal abstraction. As a consequence, the quality aspect attributed, for example, to a particular entity –both as a class and its individuals– can refer to its parts, such as the properties and/or capabilities (powers) of the entity, or the entity as a whole. To better understand the terms and relationships of the NFRs ontology, some grounding in the semantics of the foundational concepts such as Thing, Thing Category, and Assertion is needed. Ultimately, to validate the applicability of the ontological conceptualization of NFRs, three practical cases carried out in particular situations of measurement and evaluation projects are discussed, and the benefits of this ontological proposal are also highlighted.

Keywords: Non-Functional Requirements, Quality, Ontology, Particular Situation, Measurement and Evaluation Project, Multi-level Ontological Architecture.

1 Introduction

The concepts, relationships, and annotations included in the terminology of a given field of science are the foundation –as a starting point– for human understanding and the source for representing and modeling different situations of the world. The ISO 704 standard [1] states that the terminology or vocabulary of a certain field of science is not an arbitrary collection of concepts but rather a system of concepts. Therefore, the relevant concepts constitute a coherent concept system based on the relations existing among them.

Concept systems can be represented from the least to the greatest structural richness, i.e., from glossaries, taxonomies, and knowledge graphs to ontologies. While a glossary is made up of terms that designate concepts and their definitions –and optionally other annotation elements such as synonyms, acronyms, and examples, among others–, an ontology is a system of concepts with a richer model structure than a glossary. An ontology includes and graphically represents not only terms for concepts in a given field or subfield but also taxonomic and non-taxonomic relationships, constraints, and axioms. Moreover, ontologies support human understanding and learning as glossaries do, but they also provide other benefits, such as structuring and representing knowledge in a more reusable, extensible, and interoperable way, and supporting tools with semantic processing capabilities, among other features.

Particularly, the concept systems and conceptual frameworks in the subfield of science called NFRs have been the focus of research in the field of Requirements Engineering (RE) for about four decades. Note that in much RE research and practice reported in Software Engineering (SE), stakeholder needs are captured in terms of Functional Requirements (FRs) and NFRs. The concept of NFRs –a term already used in 1985 by Roman [2]– is usually called 'ilities' and implies that they must be satisfied by the existing entity or the new one to be built. In addition, as indicated by Veleda and Cysneiros [3] "*NFRs, also known as quality requirements, frequently express constraints to be imposed on functional requirements as well as general quality properties that will apply to software*". Examples of quality requirements stated as characteristics are Usability, Security, and Maintainability, among many others.

Firstly, considering the conceptual frameworks of NFRs, one of the first proposals was made in the early 90s by Chung et al. [4]. The authors provide a qualitative approach to modeling NFRs as 'softgoals', that is, goals with no clear-cut criteria for satisfaction. As commented in [5], the application of this NFR framework in practice has shown that 'softgoals' are useful for modeling early requirements elicited from stakeholders for both FRs and NFRs. Prado Leite et al. [6] also made another early framework proposal in the early 90s, which later was validated in three case studies [7]. They propose to build both the functional and the nonfunctional perspectives anchored in the Language Extended Lexicon. However, both cited contributions, particularly considering the concepts and relationships included in the respective frameworks, are not ontology-based concept systems.

Secondly, regarding the concept systems structured in ontologies for the field of RE and specifically for NFRs, since the beginning of the 2000s, there have been some formalization initiatives [8, 9]. For example, Jureta et al. [9] propose a core ontology for requirements based on the concepts of the DOLCE (*Descriptive Ontology for Linguistic and Cognitive Engineering*) foundational ontology [10]. Later, in 2015, building on this work, Guizzardi et al. [5] proposed an ontology-based interpretation of NFRs by adopting and extending the *Unified Foundational Ontology* (UFO) [11], which adheres to most DOLCE elements. Another early ontological initiative for NFRs represented by characteristics and attributes of an entity that can be measured and evaluated using metrics and indicators is documented in [12], which was expanded by Rivera et al. [13] in 2015. The latter ontology adds the concepts of quality view and cost view. However, the concepts and relationships in both works were grounded neither on a foundational ontology nor an ontological architectural framework.

The original motivation of the present research was to represent a set of interrelated concept systems structured in ontologies that integrate terms, relationships, and constraints at different levels of generality/specificity for the fields of things (entities) and assertions, processes, goals, situations, particular events, project management, functional and non-functional requirements, testing, measurement and evaluation, metrics and indicators, and so forth. These concept systems ought to be located in the context of an ontological architectural framework based on guidelines and rules for representing, reusing, and extending knowledge that should be useful for any science, not only for SE. As a practical impact, the concepts incorporated into the specifications of strategies, processes, methods, tools, etc., must emerge from these common reference terminologies.

In this direction, the main contribution of this paper is the discussion and illustration of the terms, relationships, and constraints of NFRsTDO (*Non-Functional Requirements Top-Domain Ontology*), which update and extend the [12] and [13] proposals by aligning and harmonizing them with ontological elements at core and foundational levels of a multi-tier architecture. D'Aquin et al. [14] establish a set of features and principles that must be taken into account

when designing 'beautiful ontologies'. The authors claim that the designed ontology elements should be grounded in a foundational ontology. Also, they indicate that the built ontology should be modular, extensible, and reuse already developed related ontologies, as stated by the characteristic of being modular or integrated into a modular architecture or framework. Other quality criteria that should be taken into account for the designed ontology are viz. formal simplicity and transparency promoting also the use of graphical representations for the conceptualization; coverage completeness but, at the same time, conciseness and self-intuitiveness of the elements included; and balanced representation of both taxonomic (generic and partitive hierarchical) and non-taxonomic (associative) relationships, among others.

Considering the first feature/principle commented above, NFRsTDO is based on the foundational ontology called ThingFO (*Thing Foundational Ontology*) [15]. ThingFO represents the world of any science with three generic concepts such as the particular thing (as a class or its instances), the universal thing or category of things (as a class of particular classes), and human assertions that deal with these two concepts and their relationships. Having ontologies for different application domains that inherit elements of the same core and foundational knowledge basis allows terms and relationships with similar semantics to have a shared semantic root.

Regarding the above second feature, NFRsTDO was built as an ontological component at the top-domain level that has domain-dependent concepts and relationships for the subfield of NFRs. NFRsTDO specializes terms and relationships of ontologies from foundational and core levels such as ThingFO and SituationCO, and also reuses elements from FRsTDO (*Functional Requirements Top-Domain Ontology*). All these ontologies, among others, are located in the framework of an ontological architecture called FCD-OntoArch [15], which stands for Foundational, Core, Domain, and instance Ontological Architecture. Furthermore, the NFRsTDO conceptualization is specified in UML and meets the quality criterion of having a balanced representation of types of relationships, i.e., taxonomic and non-taxonomic relationships.

To validate the applicability of NFRsTDO terms, relationships, and constraints a set of typical and practical cases will be shown for different particular situations of measurement and evaluation projects. In one case, we model NFRs in the form of a Quality Model structured in a requirements tree using the concepts of Characteristics and Attributes to specify the Maintainability characteristic [16] of a source code as an Evaluable Entity. In another case, we use the concept of Statement Item (a kind of NFR) to specify the NFRs Model with the elements of the Google Java Style Guide [17] and their mapping with Attributes of Compliance/Maintainability that can be quantified by metrics and interpreted by indicators. Additionally, we categorize and represent typical Quality Views in an SE production line with implications for evaluation and improvement strategies.

It is important to remark that the current work is an extended version of the [18] paper. Unlike this cited conference paper, in this journal article, we discuss in depth the benefits of the proposal and the problems it solves. To do so, we need to expand the description and analysis of the implemented cases, which were not previously addressed for reasons of space.

The remaining sections are organized as follows. Section 2 provides the background of FCD-OntoArch, in which the NFRsTDO ontology, among other ontological components, is located and related following a set of guidelines and rules. Section 3 analyzes the conceptualization of NFRsTDO and describes its main terms and relationships in the context of specialized concepts mainly from SituationCO and ThingFO. Section 4 illustrates the applicability of NFRsTDO terms and relationships using a set of practical cases to specify NFR models in different situations, as well as showing the usefulness of representing quality views. This section extends [18] to show the practical impact and utility of the concepts in implementing cases for measurement and evaluation projects. Section 5 discusses related work and Section 6 analyzes the benefits of the current proposal. Finally, Section 7 summarizes conclusions and future work.

2 Background of the Five-tier Ontological Architecture

As indicated in the Introduction Section, this paper presents the latest updated and harmonized version of the NFRsTDO ontology, as previously developed versions [12, 13] did not consider a clear separation of concerns and reuse of domain-independent and domain-dependent concepts and relationships when locating generic/specific elements into components in the context of a multi-tier architecture. With this goal in mind, the updated NFRsTDO component is located at the top-domain level in the context of FCD-OntoArch, as shown in Fig. 1. It should be noted that this ontological architecture was conceived in 2019 after the development of the last old version of the NFRs ontology, which was in 2015.

FCD-OntoArch is a five-tier ontological architecture that models different levels of generality/specificity that must be considered when conceiving and developing ontologies. There is a set of guidelines and rules that must be followed in the design and implementation and therefore in the placement of ontologies considering the levels of this architecture. There are two guidelines, namely:

• Guideline #1: Any ontology conceptualization/formalization developed as an artifact cannot be conceived in isolation from an explicit specification of a layered ontological architecture. For example, a foundational ontology must be found at the highest level (Foundational Ontological Level in Fig. 1) in the given architecture or framework.

• Guideline #2: At the Foundational Ontological Level of the ontological architecture, in order to fulfill the principle of completeness and conciseness of the generality of the concepts together with the principle of delegation of concerns from generality to specificity, only one foundational ontology should be found.

To identify the ontological components in FCD-OntoArch, ontologies that belong to a certain level of generality/specificity have a designation that depends on their location. Hence, an ontology at the foundational level is called Foundational Ontology (FO for short); an ontology at the core level is called Core Ontology (CO); an ontology at the top-domain level is called Top-Domain Ontology (TDO); and, an ontology at the low-domain level is called Low-Domain Ontology (LDO). Additionally, there is a layer for instances (IO). On the left side of Fig. 1, the five levels of generality/specificity are labeled. It also shows the domain-independent and domain-dependent layers. The components on the right side of the figure correspond to the ontologies already built and harmonized with ThingFO.

At the foundational level, and following Guideline #2, ThingFO is the only necessary ontology that has all three generic concepts mentioned in the Introduction Section. It represents the world through particular Things, universal Things, and Assertions. These are the unique concepts (plus their related terms and relationships) that specialize in the lower components. Consequently, terms included in lower-level ontologies must be semantically enriched with terms represented in higher-level ontologies.

At lower levels of the FCD-OntoArch architecture, the ontologies already built are: i) at the core level: ProcessCO, GoalCO, SituationCO, PEventCO (Particular Event), and ProjectCO; ii) at the top-domain level: TestTDO, FRsTDO, NFRsTDO, and MEvalTDO (Measurement and Evaluation); and iii) at the low-domain level: MetricsLDO and IndicatorsLDO.

The above ontological levels promote an accurate design within the overall architectural schema according to the following rules:

• Rule #1: Any new ontology located at level CO, TDO, or LDO of the ontological architecture represented in Fig. 1 must guarantee correspondence of its elements (terms and relationships) with the elements defined at the higher level. For example, to introduce a new ontology at the Core Ontological Level, it must be guaranteed that its elements have correspondence with the elements defined at the Foundational Ontological Level. This allows the concepts of the lower-level ontologies to be semantically enriched with the concepts of the higher-level ontologies. Additionally, non-taxonomic relationships (associations) at lower levels must have a correspondence with non-taxonomic relationships defined at higher levels.

• Rule #2: Ontologies of the same level –except at the FO and IO levels– can be related to each other, but it must be guaranteed that their joint definition (as a whole) does not violate the principles of a higher level. This implies that, if a core-level ontology uses elements from another ontology of the same level, together both models must guarantee a correct definition with respect to the foundational ontology. This allows the terms and relationships of the ontologies of the same level to complement each other, maintaining correspondence with the definitions of the ontologies of the higher levels.

• Rule #3: At the Instance Ontological Level, only individuals of particular terms can be found. A particular concept (concrete/particular class) represented at the foundational level by the term Thing, or any of its subclasses at lower levels results in instances. Therefore, an individual is an instance of a particular class at higher levels.

At this point, for instance, what does it mean that the concepts of an ontology at one level can be semantically enriched by other higher-level ontology concepts? It means, according to Rule #1, that the concepts (usually designated by terms) of an ontology at a certain level (e.g. top-domain or core) can reuse or 'inherit' the semantics of the corresponding more general concepts. For example, in Fig. 2, the term NFRs Model in NFRsTDO is stereotyped with the term Artifact, which is a core term in ProcessCO [19]. That is, NFRs Model is an Artifact, or inherits the semantics of, or is semantically enriched by the term Artifact. If we were to examine the ProcessCO ontology, we could see that the term Artifact is a kind of Work Product and, in turn, the term Work Product is semantically enriched by the term Thing of ThingFO.

The definition of the term NFRs Model must indeed be defined according to the RE field, but the designer already has the advantage of knowing the holistic meaning of the term Artifact, as it is explicitly defined in the ProcessCO ontology. In other words, these core terms provide a semantic basis and reference that give support to better defining and understanding more specific domain terms.

Lastly, Fig. 1 shows with red dashed lines the links from NFRsTDO to other components located in the architecture. Looking at the dependency relationships of NFRsTDO with other higher ontological components, some of its terms are semantically enriched with terms from SituationCO and ProcessCO. Also, NFRsTDO has a dependency relationship with the term Assertion, which is represented in ThingFO. Furthermore, at the top-domain

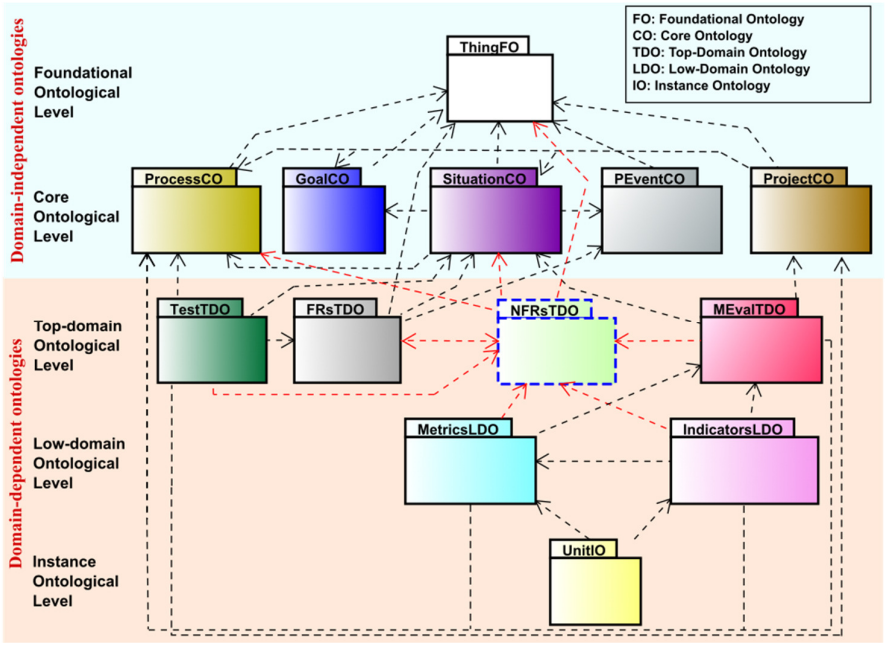

Figure 1: Location of the NFRsTDO component or module in the context of the five-tier ontological architecture called FCD-OntoArch. Note that NFRs stands for Non-Functional Requirements, FRs stands for Functional Requirements, and MEval for Measurement and Evaluation

level, we observe that the components MEvalTDO and TestTDO depend on some terms of NFRsTDO, and there is also a double link between FRsTDO and NFRsTDO. Finally, MetricsLDO and IndicatorsLDO reuse or extend some terms from NFRsTDO.

3 The Non-Functional Requirements Ontology in the Context of FCD-OntoArch

The conceptualization of the NFRsTDO component, which includes the NFRs_view subcomponent, is depicted at the bottom of Fig. 2. A fragment of the SituationCO ontology is shown in the center of the figure and a fragment of ThingFO is represented at the top of the figure. Note that all ontological conceptualizations are specified in UML. In numerical terms, NFRsTDO includes 15 top-domain terms –most of them semantically enriched with extended terms from two core ontologies such as SituationCO and ProcessCO, and with specialized terms from ThingFO, but one was reused from FRsTDO–, 13 taxonomic relationships, and 12 non-taxonomic relationships defined between terms, in addition to 18 properties and constraints.

It is important to note that the ontological components located in this architecture use stereotypes as a mechanism to semantically align terms. Concerning the procedural way to enrich a given term from a higher-level term, Becker et al. [19] argued that stereotypes are a more appropriate mechanism than inheritance relationships, as they generate a loose coupling between a lower-level component and a higher-level component. Consequently, stereotypes can reduce the complexity of the specification, thereby promoting understandability and communicability.

Next, in subsection 3.1, we first describe the three generic concepts of ThingFO, which become more specific at lower levels as discussed in Section 2. Then, in subsection 3.2, a subset of key terms and relationships from SituationCO are outlined. Finally, in subsection 3.3, we discuss in depth the terms, properties, relationships, and constraints of the NFRsTDO ontology.

3.1 The Three Key Concepts of ThingFO

As mentioned above, a foundational ontology is a domain-independent model of the most key generic concepts such as those specified at the top of Fig. 2.

On the one hand, the term Thing represents a particular, tangible or intangible, perceivable or conceivable entity in the modeled world, but not a universal category, which is represented by the term Thing Category. The Particular Concept named Thing represents and implies unique individuals or instances. It is modeled by the concrete UML class and supports subtyping by inheritance at lower layers, except at the Instance Ontological Level, according to Rule #3.

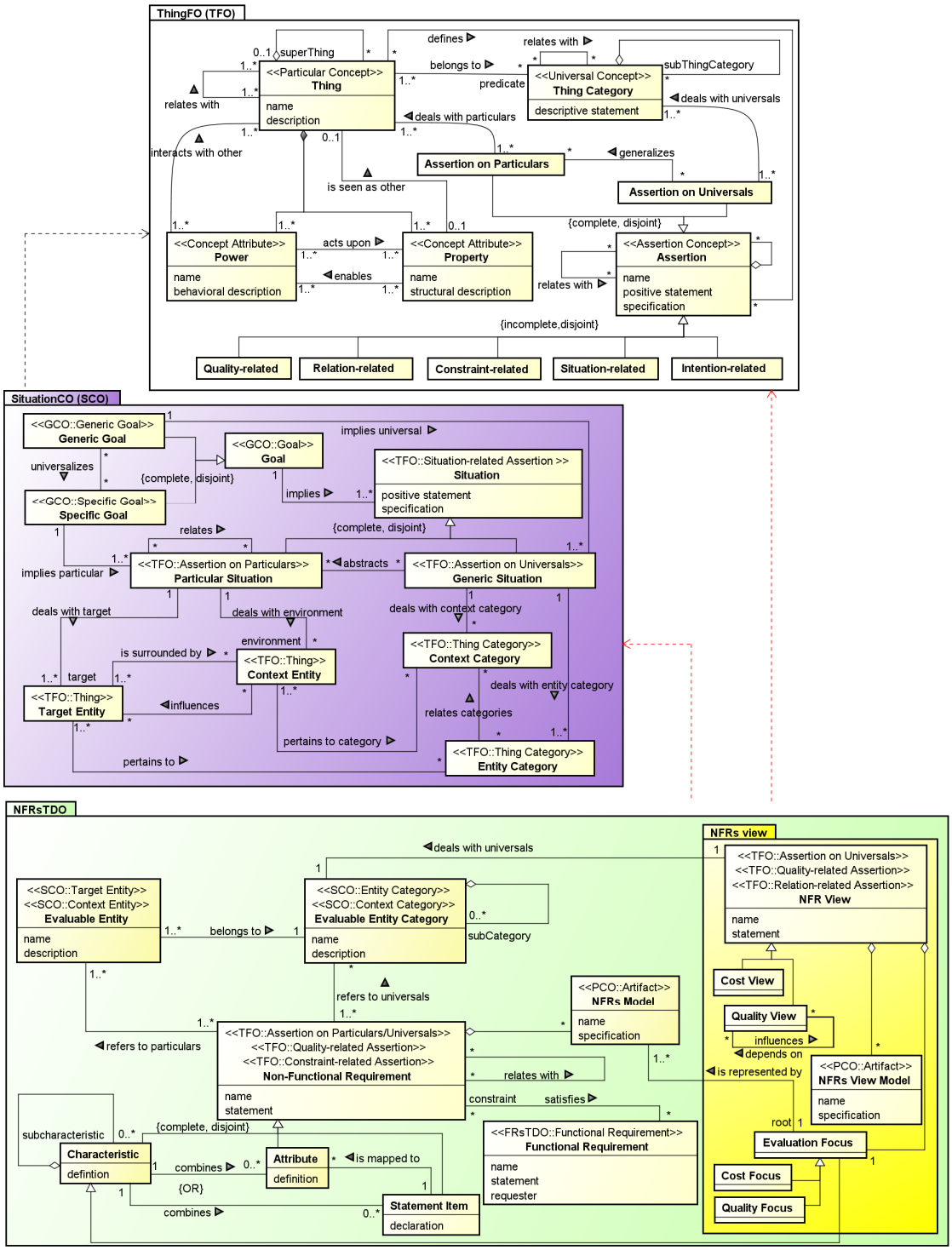

Figure 2: At the bottom, are shown all the terms and relationships of the Non-Functional Requirements Top-Domain Ontology (NFRsTDO). This ontology includes the NFRs_view subontology (yellow component). In the middle, a fragment of the Situation Core Ontology (SituationCO). At the top, a fragment of the Thing Foundational Ontology (ThingFO). Note that TFO stands for Thing Foundational Ontology, PCO for Process Core Ontology, GCO for Goal Core Ontology, PjCO for Project management Core Ontology, SCO for Situation Core Ontology, and FRsTDO for Functional Requirements Top-Domain Ontology

For example, a class Car (or Airplane, Motorcycle, Boat, etc.) is a type of Vehicle. Furthermore, the latter is a kind of Product, which in turn has the semantics of Thing. Therefore, there are instances of each class at the Instance Ontological Level since they are individual cars, airplanes, or boats.

Moreover, a Thing is not a particular entity without its Properties and its Powers. A Property refers to the intrinsic constitution, structure, or parts of a Thing, while a Power (Capability) refers to what a Thing does, can do, or behaves. A Thing with its Properties and Powers all emerge simultaneously to form a unity, so all three terms are necessary and sufficient for the existence of this unity as a concept [20]. A Car is not a car without its Properties (e.g. engine, wheels, seats, etc.) and its Powers (e.g. accelerate, turn right or left, brake, etc.).

On the other hand, the Universal Concept designated by the term Thing Category in Fig. 2 is useful for classification and abstraction purposes. Whereas a Thing is modeled by the concrete class, which implies unique individuals or instances, a Thing Category is modeled by the abstract or categorical class (class of classes) in which the instances do not have the valuable meaning of an individual. On the contrary, it can only be mentally formed or developed by human beings as an abstract construct, which in turn, hierarchies of partitive sub-categories can be represented. As a consequence, a Thing Category does not exist, is, or can be in a given particular world as individuals of a Thing do, but rather Things belong to the intended Thing Category. For the above example, Car and Motorcycle belong to the Ground Transport category, which is a subcategory of the Transport Category by Physical Means.

Lastly, the third key concept in ThingFO is Assertion. This term has a great conceptual impact when a human agent intentionally represents and models Particular and Universal Things and Situations of the world at hand. The term Assertion is defined in [15] as a "*Class or type that represents a positive and explicit statement or expression that somebody makes about something concerning Things, or their categories, based on thoughts, perceptions, facts, intuitions, intentions and/or beliefs, conceived with an attempt to provide current or subsequent evidence*". To be valuable, actionable, and ultimately useful for any science, an Assertion should largely be verified and/or validated by theoretical and/or empirical evidence. Assertions can be expressed by informal, semi-formal, or formal specification languages. Thus, a specification can include textual expressions in natural language, mathematical and logical expressions, or well-formed models/diagrams, among other representations.

There is a term called Assertion on Particulars for Things, and another called Assertion on Universals for Thing Categories. Fig. 2 shows five subtypes of Assertions, two of which are: i) Quality-related Assertion that deals with requirements to be specified regarding quality aspects of a Thing and possibly related entities, the satisfaction of which can be evaluated; and, ii) Constraint-related Assertion which deals with the specification of restrictions or conditions imposed on concept elements, relationships, and interactions that must be satisfied or evaluated to be true in given situations or events.

3.2 Some Key Concepts of SituationCO

A fragment of the SituationCO component is depicted in the center of Fig. 2. This ontology [15] considers core (domain-independent) terms and relationships for Particular and Generic Situations to a given intentionality at hand. The term Situation is defined as "*It is a Situation-related Assertion that explicitly states and specifies the combination of circumstances, episodes, and relationships/events embracing particular entities and their surroundings, or categories of entities and their related generic context, which is of interest and relevant to be represented by a human agent/organization with an established Goal*".

Most of the SituationCO terms are derived directly from ThingFO, but it also borrows some core terms from the GoalCO and ProcessCO ontologies as suggested in Fig. 1. For example, the terms Goal, Specific Goal, and Generic Goal are reused from GoalCO and all have the semantics of Intention-related Assertion, which is a term that comes from ThingFO. The term Thing semantically enriches the terms Target Entity and Context Entity as shown in Fig. 2. The term Thing Category semantically enriches the terms Entity Category and Context Category. It is also represented in the figure that one or more concrete Target Entities pertain to the Entity Category, whereas Context Entities pertain to category Context Category. Notice that the pertains to relationship, is a specialization of that named belongs to from ThingFO.

It is worth remarking that foundational and core terms and relationships are specialized, and conceptual blocks or patterns are reused by top-domain and low-domain ontologies. For example, MEvalTDO is a top-domain ontology for Measurement and Evaluation activities, methods, and projects, which is terminologically benefited from the concepts of ThingFO, SituationCO, ProcessCO, and ProjectCO. The terms Particular Situation, Target Entity, and Context Entity included in SituationCO semantically enrich the term Evaluation Particular Situation and its related terms, as well as their corresponding relationships are specialized. Therefore, MEvalTDO specializes the Particular Situation conceptual block or pattern, as we will see in Section 4.

Returning to the example of subsection 3.1, let's suppose a customer needs to understand the performance of a concrete car on two types of roads. In this Evaluation Particular Situation, the Evaluable Target Entity is the Car, and

the Evaluable Context Entities are for instance a Paved Road and a Dirt Road. Then, the Evaluation Project operationalizes the Specific Goal purpose 'understand' by deriving an NFR of Performance that refers to the Car in both contexts. In other words, in any Particular Situation of any represented problem, one or more entities (Things) in the role of the `target` is/are surrounded by none or many entities (Things) in the role of the `environment`.

3.3 The Non-Functional Requirements Ontology at the Top-Domain Level called NFRsTDO

The subsections summarized above may help the reader better understand the concepts and relationships of NFRsTDO and, along with some MEvalTDO concepts, their illustration in practical cases in Section 4. Next, we first analyze the terms Evaluable Entity, Non-Functional Requirement, Characteristic, Statement Item, Attribute, and NFRs Model, as well as their relationships shown at the bottom of Fig. 2.

The term Evaluable Entity is a Target or Context Entity that represents a concrete entity or object to be evaluated. Note that depending on a given evaluation project and situation, an Evaluable Entity has the semantics of a Target Entity or Context Entity, which in turn, as mentioned above, are Things. An example of an Evaluable Entity, as we will illustrate in Section 4, is a specific Java Program, that is, the source code written in the Java programming language. Other examples of Evaluable Entities are specific products (a car, road, document, model, smartphone device, etc.), systems, resources, work processes, services, organizations, etc.

NFR is a Quality-, Constraint-related Assertion (both Particular and Universal Assertion) that specifies an aspect in the form of a Characteristic, Attribute, or Statement Item to be evaluated on how or how well an Evaluable Entity performs or shall perform. The qualitative aspect of the Evaluable Entity, which in the end is a particular Thing, can refer to its Properties, Powers, or the entire Entity.

Characteristic (see synonyms in Table 1) is a (kind of) NFR that represents an evaluable, non-elementary aspect attributed to a particular entity or its category by a human agent. Note that a Characteristic can be evaluated but cannot be measured as an Attribute –at least in a non-very trivial way such as "good" or "bad". Examples of it are Performance, Maintainability, Usability, and Comfort, among many others.

Attribute is a (kind of) NFR that represents a measurable and evaluable physical or abstract aspect attributed to a particular entity or its category by a human agent. Note that an Attribute is a measurable and evaluable elementary NFR, i.e., an elementary quality to be quantified. For example, if Comfort is the name of a Characteristic of a Car, an Attribute combined with it can be named Amount of Seats, among others.

Statement Item –or Item for short– is a (kind of) NFR that represents a declared textual expression of an evaluable physical or abstract aspect asserted for a particular entity or its category by a human agent. Note that an Item can be for instance an assertive element in a questionnaire instrument, an assertive element in a heuristic checklist, or an assertive element in a programming style guide, as we will illustrate in the next Section.

Fig. 2 also shows that the terms Characteristic, Attribute, and Statement Item are the unique concepts needed to specify an NFR using the `complete` and `disjoint` constraints. Additionally, both a Characteristic and an Attribute have a definition, while an Item has a declaration. All of them also have a name as a concept attribute respectively.

NFRs Model has the semantics of Artifact (from ProcessCO) that specifies and represents Non-Functional Requirements. Notice that any model in science is the resulting artifact of an Assertion specification. As an example of a quality model, see the note for this term in Table 1.

Taking into account the relationships for the above terms, Fig. 2 shows that an NFR relates with none or many NFRs, where an NFR always refers to particulars i.e., an Evaluable Entity. (Note that the relationship refers to particulars is a specialization of the deals with particulars one from ThingFO). In addition, a Characteristic is a non-elementary NFR, which combines Attributes or Statement Items. Besides, a Statement Item, which is an NFR as well, is mapped to none or many Attributes. This latter relationship will be illustrated in a practical case in subsection 4.2.

In our conceptual framework, as in other concept systems and frameworks commented in the Introduction Section, there is an explicit link between an NFR and an FR –see the FR term definition and notes in Table 1. In particular, we can say that an NFR satisfies none or many FRs. In the other direction, as specified in the FRsTDO component, we can say that an FR requires to be satisfied by none or many NFRs in the role of constraint.

Finally, for the NFRs_view subcomponent, let us primarily discuss the terms NFR View, Evaluation Focus (in particular the terms Quality View and Quality Focus), and Evaluable Entity Category.

NFR View has the semantics of Assertion on Universals from the ThingFO ontology, and more specifically, it has the semantics of Quality-related Assertion and Relation-related Assertion. The term NFR View relates one Evaluable Entity Category to one Evaluation Focus.

Evaluation Focus is a Characteristic (which, in turn, is an NFR with Assertion semantics) that represents the `root`, i.e., the most generic quality/cost aspect of an NFRs Model. Fig. 2 shows that a Quality Focus is an Evaluation Focus.

Table 1: Definition of most of the terms of NFRsTDO and some notes. All definitions are available in [21]

NFRs Term	Definition
Attribute (synonym: *Property; Elementary Aspect*)	It is an NFR that represents a measurable and evaluable physical or abstract aspect attributed to a particular entity or its category by a human agent. Note: An Attribute can be quantified by employing metrics and interpreted utilizing elementary indicators, which are concepts included in the MetricsLDO and IndicatorsLDO ontologies.
Characteristic (synonym: *Dimension; Factor; Non-elementary Aspect; Calculable Concept; Evaluable Concept*)	It is an NFR that represents an evaluable, non-elementary aspect attributed to a particular entity or its category by a human agent. Note 1: A Characteristic can have sub-characteristics. Note 2: A Characteristic can be calculated and interpreted utilizing derived indicators and aggregation functions, which are concepts included in the IndicatorsLDO ontology.
Evaluable Entity (synonym: *Evaluable Particular Entity; Evaluable Particular Object*)	It is a Target Entity or Context Entity that represents a particular (concrete) entity to be evaluated.
Evaluable Entity Category synonym: *Evaluable Universal*)	It is an Entity Category or Context Category to which particular Evaluable Entities belong to. Note: A universal (or category) can be classified into Evaluable Entity Category, Developable Entity Category, Testable Entity Category, Observable Entity Category, etc., depending on the intention of the human agent and situation.
Functional Requirement (acronym: FR)	It is an Assertion on Particulars that specifies what the existing Developable Entity (or sub-Developable Entity) does, or the new one shall do by referring to its features and/or capabilities considering a given requester's need. Note 1: A Functional Requirement has the semantics of Assertion on Particulars from the ThingFO ontology, and more specifically it has the semantics of one or more assertions such as Structure-, Behavior-, Action-, Relation-related Assertions depending on the development Particular Situation. Note 2: A Functional Requirement (term reused from the FRsTDO ontology) can require to be satisfied by constraints or 'ilities', which are stated by NFRs.
Non-Functional Requirement (acronym: NFR)	It is a Quality-, Constraint-related Assertion that specifies an aspect in the form of a Characteristic, Attribute, or Statement Item to be evaluated on how or how well an Evaluable Entity performs or shall perform. Note: The term NFR has the semantics of both Particular and Universal Assertion.
NFRs Model (synonym: *Quality Model*)	It is an Artifact that specifies and represents Non-Functional Requirements. Note: For example, the structure of the NFRs Model for quality represented in the ISO 25010 standard [16], hierarchically models a set of Characteristics, sub-characteristics, and Attributes (or properties as regards the used term in it), as well as the relationships between them, which provide the basis for specifying the NFRs and their further evaluation for software products and systems.
Statement Item (synonym: *Item; Guideline; Heuristic*)	It is an NFR that represents a declared textual expression of an evaluable physical or abstract aspect asserted for a particular entity or its category by a human agent. Note: A Statement Item can be quantified by using the scales designed in a questionnaire instrument, or utilizing other qualitative ways. Also, it can be mapped to Attributes as shown in Fig. 2.
NFRs view Term *(fragment of the terms included)*	
Evaluation Focus	It is a Characteristic that represents the root of an NFRs Model.
NFR View (synonym: *NFR Perspective*)	It is an Assertion on Universals that relates one Evaluable Entity Category to one Evaluation Focus. Note 1: The Evaluable Entity Category must be the super-category, i.e., the highest abstraction level of an Entity Category of value to be evaluated in a given project/organization. Note 2: The Evaluable Focus must be the root Characteristic, which represents the Characteristic with the highest level of abstraction in an NFRs Model of value to be evaluated in a given project/organization.
NFRs View Model	It is an Artifact that specifies and represents NFR Views. Note: An NFRs View Model has the semantic of Artifact, which is a term coming from the ProcessCO ontology.
Quality Focus	It is an Evaluation Focus for quality.
Quality View (syn.: *Quality Perspective*)	It is an NFR View for quality.

Examples of names of Quality Focuses are viz. Resource Quality, Work Process Quality, Internal Quality, External Quality, and Quality in Use, among others.

The other term that includes the definition of NFR View is Evaluable Entity Category. It is an Entity Category or Context Category to which particular Evaluable Entities belong to. Depending on a given Situation, an Evaluable Entity Category has the semantics of Entity Category or Context Category, which in turn both have the semantics of Thing Category. Examples of names of Evaluable Entity Categories are viz. Software Product Category, Software System Category, among many others.

A kind of NFR View is the Quality View whose sub-categories names are, for example, Software Product Quality View, Software System Quality View, etc. Thus, the Software Product Quality View has the Internal Quality focus and the association (deals with universals) with the Software Product Category. Furthermore, there are two relationships between Quality Views shown in Fig. 2, namely: influences and depends on. We can state, for instance, that the Software System Quality View depends on the Software Product Quality View or that the latter influences the former. The representation of these relationships between sub-categories is paramount when designing evaluation strategies and patterns as discussed in [13].

Next, we illustrate three practical cases to observe and analyze the impact of these concepts in three typical situations of measurement and evaluation projects. The third case is devoted to showing an NFRs View Model (see the definition of this term in Table 1) for quality, as well as evaluation strategies for different NFR Views.

4 The Concepts and Relationships of NFRsTDO in Practice

As discussed in Section 2, at the top-domain level, we see in Fig. 1 that the MEvalTDO and TestTDO components depend on elements of NFRsTDO. Also, both extend some terms from SituationCO, ProcessCO, and ProjectCO.

In particular, the MEvalTDO ontology deals with Measurement and Evaluation (ME) activities and methods in general, while MetricsLDO and IndicatorsLDO are low-domain level ontologies, addressing more specific ME activities and methods based on metrics and indicators, respectively. To illustrate the applicability of NFRsTDO concepts, it is necessary to first show a subset of MEvalTDO terms and relationships. This is because for an organization to address an ME problem/situation and organize work into projects, terms such as Evaluation Project, Evaluation Goal, etc., are not represented in NFRsTDO just to maintain conciseness and separation of concerns between components.

Fig. 3 shows an excerpt from the conceptualization of MEvalTDO. Note that its full conceptualization, definitions, and axioms are in [22]. The figure exhibits that an Evaluation Project operationalizes an Evaluation Goal, which is derived in one or more ME NFRs (see the term ME Non-Functional Requirement). This last term is enriched semantically from NFRsTDO, as well as the terms Evaluable Target Entity and Evaluable Context Entity, whereas the term NFRs Model is completely reused. In addition, the reader can see that the term Particular Situation of SituationCO (representing a conceptual block or pattern that includes the terms Target Entity and Context Entity and their relationships) is specialized here in Evaluation Particular Situation. Lastly, the conceptual model also specifies that an Evaluation Project Process has an Evaluation Process, which in turn associates an Evaluation Strategy. An Evaluation Strategy is a project resource that helps achieve a given Evaluation Goal.

Aspects of three practical cases of measurement and Evaluation Projects are discussed below. In subsection 4.1, we represent NFRs using the 'Maintainability' Characteristic and its Attributes for a Java Program, as an Evaluable Target Entity. To analyze the impact of these concepts in practice, an indirect metric and elementary indicator for an Attribute are illustrated in addition to the values achieved. In subsection 4.2, we specify NFRs using the concepts of Statement Items of the Google Java Style Guide and their mapping with 'Maintainability' Attributes. Additionally, we discuss the Define NFRs activity represented in an Evaluation Strategy. Finally, in subsection 4.3, we represent typical Quality Views (QVs) on an SE production line of interest for evaluation, as well as examples of Evaluation Strategies that deal with one and two QVs.

4.1 Illustrating Practical Case I

This practical case was conducted in an academic context in 2023 and published at the CoNaIISI national conference on Information Systems, so the full paper written in Spanish is available in [23].

The *ME statement* (see this property of the term Evaluation Goal in Fig. 3) is "to understand and improve 'Maintainability' and its 'Analyzability' and 'Testability' sub-characteristics for the 'GUICalculator.java v1.0' source code in the context of an advanced Systems Engineering course". This goal has an *ME purpose* of 'understanding' and 'improving' the Java source code. It also implies an Evaluation Particular Situation whose Evaluable Target Entity –with the semantics of Evaluable Entity from NFRsTDO– is a Java Program.

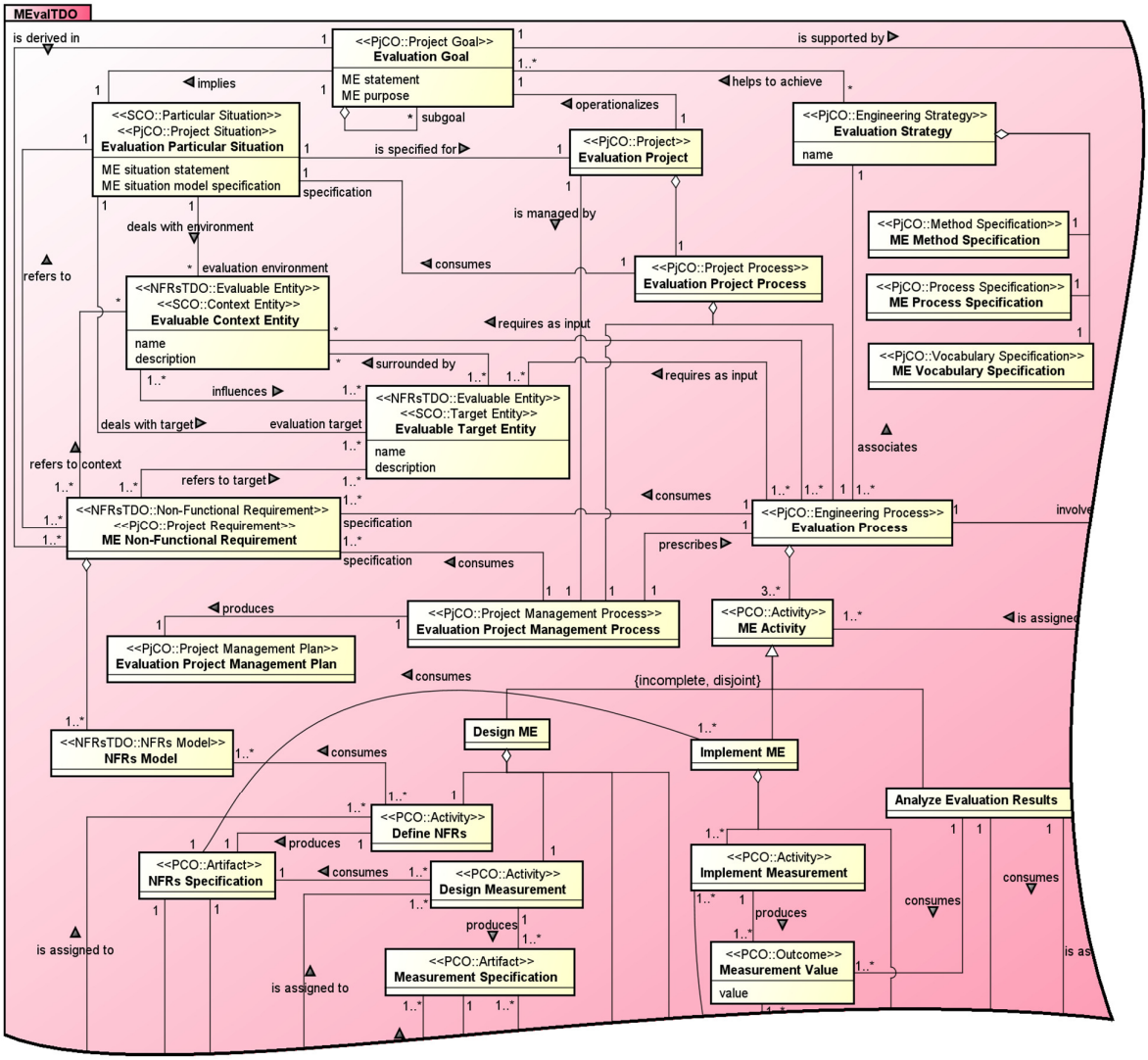

Figure 3: Extract from the conceptualization of MEvalTDO. The term Particular Situation of SituationCO is made more specific namely Evaluation Particular Situation. Also, the semantics of a pair of terms from the NFRsTDO ontology are reused. Note that PjCO stands for Project management Core Ontology, PCO for Process Core Ontology, and SCO for Situation Core Ontology

Note that 'GUICalculator.java v1.0' is an individual or instance of the concrete class named Java Program in this Evaluation Project. It is also important to note that for this practical case, it was not necessary to establish an Evaluable Context Entity. A reason is that the Quality Focus is Internal Quality, which is the `root` Characteristic of the Quality Model represented in Table 2. Instead, if the Quality Focus were, e.g. Quality in Use, then in this Evaluation Particular Situation the specification of the context is mandatory, as documented in [24].

According to the above *ME statement*, the Evaluation Goal is derived in NFRs such as the Characteristic called 'Maintainability' and two of its sub-characteristics named 'Analyzability' and 'Testability', which are quality aspects included in the model of the ISO 25010:2011 standard [16]. Note that the updated ISO 25010 was issued in late 2023, but the *names* of the 'Maintainability' sub-characteristics remain unchanged.

In addition to a *name*, an NFR as a Characteristic or Attribute has a *definition* as specified in Fig. 2. We have reformulated the *definition* of 'Maintainability' of [16] as the "*Degree to which a product or system can be diagnosed and modified by maintenance personnel*". Likewise, we have defined 'Analyzability' as the "*Degree to which the understandability, deficiencies, or causes of failure of a product can be diagnosed, or parts of a product can be identified for modification considering the impact of their changes on the entire product or system*" and 'Testability'

Table 2: The NFRs tree (Quality Model) specification for the 'Maintainability' Characteristic and its sub-characteristics and Attributes in Case I

Characteristics and Attributes (*in italics*)
1. Maintainability
1.1. Analyzability
1.1.1. Code understandability
1.1.1.1. Control flow understandability
1.1.2. Documented code availability
1.1.2.1. Documented class availability
1.1.2.2. Documented method availability
1.2. Testability
1.2.1. Code coverage complexity
1.2.1.1. Control flow complexity

as "*Degree to which test criteria can be established for a system, product, or component and tests can be performed to determine whether these criteria have been met*".

The 'Analyzability' sub-characteristic has in turn a sub-sub-characteristic that combines the Attribute named 'Control flow understandability' (coded 1.1.1.1 in the NFRs tree of Table 2). It is defined as the "*Degree to which the control flow of a program is easily understood by developers and other related professionals, which implies that the order of execution of instructions, conditions, and nesting within the code is understandable*".

In this case, 'Control flow understandability' is quantified using a metric named 'Density of Methods with high Cognitive Complexity', which is based on the measurement procedure of the 'Cognitive Complexity' metric [25]. For 'Testability', the Attribute named 'Control flow complexity' (coded 1.2.1.1 in Table 2) is quantified using a metric based on the 'Cyclomatic Complexity' [26] measurement procedure. Currently, both metrics are widely used by quality assurance professionals in the software industry, mainly supported by the SonarQube (https://www.sonarsource.com/products/sonarqube/) tool.

To observe the practical impact of these concepts, in particular, for the NFR and Attribute terms from the measurement and evaluation standpoint, we illustrate the quantification of the 'Control flow understandability' Attribute by specifying an indirect and a direct metric, as well as an elementary indicator and the values obtained for the source code 'GUICalculator.java' in version 1.0 and the improved version (v1.1).

As commented above, to quantify the attribute 'Control flow understandability', the indirect metric 'Density of Methods with high Cognitive Complexity' (DMhCoCo) was defined. The specification of this indirect metric is shown in Table 3.

Table 3: Specification of the indirect metric to quantify the 'Control flow understandability' (1.1.1.1) Attribute

Indirect Metric name: Density of Methods with high Cognitive Complexity (DMhCoCo) **Quantified Attribute name:** *Control flow understandability* (*1.1.1.1*)
Objective: Determine the proportion of methods with high cognitive complexity considering all Java classes and files. **Author:** Pablo Becker and Luis Olsina **Version:** 1.0
Calculation Procedure **Specification / Formula:** $$\boldsymbol{DMhCoCo} = \frac{\sum_{i=1}^{\#JF}\sum_{j=1}^{\#C_i} \#\boldsymbol{MhCoCo_{ij}}}{\sum_{i=1}^{\#JF}\sum_{j=1}^{\#C_i} \#\boldsymbol{M_{ij}}}$$
Scale: Numeric **Scale Type name:** Ratio **Value Type:** Real **Representation:** Continuous
Unit **Name:** Methods with high cognitive complexity over total methods **Acronym:** MhCoCo/M
Related Metrics: Number of Java Files (#JF) Number of Classes in the file (#C) Number of Methods in the class (#M) Number of Methods in the class with high Cognitive Complexity (#MhCoCo)

In addition, for each indirect metric one or more direct metrics were specified. For DMhCoCo, four direct metrics were defined, namely: 'Number of Java Files' (#JF), 'Number of Classes in the file' (#C), 'Number of Methods in the class' (#M), and 'Number of Methods in the class with high Cognitive Complexity' (#MhCoCo). Table 4 shows the specification of the direct metric #MhCoCo, which counts the number of methods that have a high cognitive complexity, i.e. a score higher than 15. This threshold value has been used since the SonarQube tool defines by default that methods should not have a cognitive complexity higher than 15. High cognitive complexity means that programmers who must read the code to modify it must make a significant cognitive or mental effort to understand the code flow. The #MhCoCo measurement (count) procedure is based on the following rules [25]: 1) code is considered more complex when it uses structures that allow multiple statements to be readably shorthanded into one; 2) code is considered more complex for each break in the linear flow of the code; and 3) code is considered more complex when flow-breaking structures are nested.

Since in our approach (Evaluation Strategy) a measurement value does not represent the level of satisfaction of an Attribute, a transformation must be performed to convert the measurement value (data) into a new value (information) that can be interpreted. Therefore, for each Attribute of Table 2, an elementary indicator was specified. As an example, Table 5 shows the specification of the elementary indicator named 'Performance level in Control Flow Understandability' (P_CFU) that helps evaluate the attribute 'Control flow understandability' (1.1.1.1). This indicator, like the rest of the elementary indicators designed, has three levels of acceptability considering the percentage unit of the indicator scale: 'Unsatisfactory' (values lower than 80%), 'Marginal' (values equal to or greater than 80% and less than 100%) and 'Satisfactory' (values equal to 100%). Thus, the 'Unsatisfactory' acceptability level conveys that

Table 4: Specification of one of the Direct Metrics related to the "Density of Methods with high Cognitive Complexity" (DMhCoCo) metric

Direct Metric name: Number of Methods in the class with high Cognitive Complexity (#MhCoCo)
Objective: Determine the number of methods with high cognitive complexity found in a class of a Java file. **Author:** Pablo Becker and Luis Olsina **Version:** 1.0
Measurement Procedure **Type:** Objective **Specification:** list1 = {if, else if, else, ternary operator, switch, for, foreach, while, do while, catch, sequences of binary logical operators, each method in a recursion cycle} list2 = {if, ternary operator, switch, for, foreach, while, do while, catch} list3 = {if, else if, else, ternary operator, switch, for, foreach, while, do while, catch, nested methods and method-like structures such as lambdas} #MhCoCo = 0 For each method declaration in the class CoCo = 0 NL = 0 For each line of code in the method For each statement/keyword that is included in list1 CoCo = CoCo + 1 If there is a statement that is included in list2 then: CoCo = CoCo + NL If there is a statement that is included in list3 then: NL = NL + 1 If a line of code includes the closing of the code block started by an element of list3 then: NL = NL – 1 if CoCo > 15 then: #MhCoCo = #MhCoCo + 1 Note 1: In this metric, constructors (their declaration) are considered methods. Note 2: A method's declaration defines its signature, name, parameters, and return type.
Scale: Numeric **Scale Type name:** Absolute **Value Type:** Integer **Representation:** Discrete
Unit **Name:** Method with high cognitive complexity **Acronym:** MhCoCo

Table 5: Specification of the Elementary Indicator for the Attribute 'Control flow understandability' (1.1.1.1 in Table 2). Note: DMhCoCo represents the metric 'Density of Methods with high Cognitive Complexity'

Elementary Indicator name: Performance level in Control Flow Understandability (P_CFU) **Evaluated Attribute name:** *Control flow understandability* (*1.1.1.1*)
Author: Pablo Becker and Luis Olsina **Version:** 1.0
Elementary Function **Specification:** P_CFU = (1 – DMhCoCo) * 100 **Decision Criteria (Acceptability Levels):** **Name:** ■ Unsatisfactory **Description:** [0;80) **Description:** Indicates that corrective actions must be carried out with high priority. **Name:** ◆ Marginal **Description:** [80;100) **Description:** Indicates that corrective actions should be performed. **Name:** ● Satisfactory **Range:** [100;100] **Description:** Indicates that corrective actions are not necessary since the attribute meets the required quality satisfaction level.
Scale: Numeric **Scale Type name:** Ratio **Value Type:** Real **Representation:** Continuous
Unit **Name:** Percentage **Acronym:** %

corrective actions should be carried out with high priority while the 'Marginal' acceptability level indicates that corrective actions could be carried out. We decided to use the traffic light metaphor to facilitate the visualization of the levels of satisfaction achieved: ■ red / 'Unsatisfactory', ◆ yellow / 'Marginal', and ● green / 'Satisfactory'.

In summary, considering the abovementioned concepts and relationships included in the NFRsTDO ontology, for example, an Attribute is a kind of NFR that represents a measurable and evaluable physical or abstract aspect attributed to an entity by a human agent. It is, therefore, an elementary qualitative requirement of an entity –i.e. a Quality-related Assertion that deals with an aspect of an entity such as a Thing– to be quantified by metrics and interpreted by an elementary indicator. In this way, once the measurement and evaluation have been carried out, the indicator's value indicates the level of satisfaction achieved by the Attribute –as an elementary non-functional requirement to be satisfied.

For the running example, Table 6 shows the values obtained for the indirect metric and the values calculated for the elementary indicator. Regarding the evaluated Attribute 1.1.1.1, with a score of 83.33% (◆), it was recommended to "Modify (or replace) methods that have high cognitive complexity so that they are easier to understand and therefore to maintain". Based on this recommendation, after making the change in the source code and performing the re-evaluation in 'GUICalculator.java v1.1', the P_CFU indicator reached 100% (●).

Therefore, in pursuing the stated *ME purpose* of 'understanding' and 'improving', this increase in the indicator value implies an enhancement in this specific aspect of the Evaluable Target Entity. It is worth noting that the complete case documented in Spanish is available at https://bit.ly/CoNaIISI23_Apendices.

Table 6: Fragment of the NFRs tree with the measurement values for 1.1.1.1 including the evaluation values of Elementary/Derived Indicators (EI/DI) obtained for versions 1.0 and 1.1 of the 'GUICalculator.java' code. Note: The symbol ● means 'Satisfactory'; ◆ 'Marginal', and ■ 'Unsatisfactory'

Characteristic and Attribute (*in italic*)	**v1.0**		**v1.1**	
	measurement value	**EI/DI value** [%]	**measurement value**	**EI/DI value** [%]
1. Maintainability	-	58.33 ■	-	100 ●
⋮	⋮	⋮	⋮	⋮
1.1.1.1. Control flow understandability	0.166	83.33 ◆	0	100 ●
⋮	⋮	⋮	⋮	⋮

As a final comment, in an Evaluation Project, we have to identify these elementary qualitative aspects by giving them an explicit *name* and *definition*. Since an Attribute can in principle be quantified by one or more metrics, one issue is the *name* of the Attribute and the other is the name of the metric. This concern is often neglected in the measurement literature, so the metric name is frequently the same as the Attribute *name*.

4.2 Illustrating Practical Case II

Like the case mentioned above, this one was also conducted in an academic context in 2022, but the chapter and supplementary material were published in English [27]. In this practical case, the *ME statement* of the Evaluation Goal is "to understand and improve Compliance of the 'GUICalculator.java v2.0' source code with a subset of Items of the Google Java Style Guide in the context of an advanced Systems Engineering course". This goal implies an Evaluation Particular Situation whose Evaluable Target Entity is a Java Program, particularly, the 'GUICalculator.java v2.0' instance.

On the one hand, a Java Program as an Evaluable Entity is an Artifact and, therefore, a particular Thing. It has Properties such as class, file, and documentation elements. In turn, a class has method and field members expressed by statements or instructions. As Powers or capabilities of this static (inanimate) object, we can represent a list of behaviors such as 'can_be_read', 'can_be_modified', etc. Note that Property and Power are terms that represent the two key Concept Attributes that a Thing must have, as shown in Fig. 2.

On the other hand, the Items of the Google Java Style Guide deal with elementary quality aspects that a Java Program must adhere to or comply with. The Items or Guidelines chosen for this practical case are shown in the 1st column of Table 7. Each Item has a *name* and a *declaration* according to [17]. For example, in Table 7, the originally coded element 3.4.1 in [17] has the *name* 'Exactly one top-level class declaration' and its *declaration* is "Each top-level class resides in a source file of its own". Furthermore, the 2nd column of Table 7 shows in italics all the Attributes identified for this study and their correspondence with the Items of the Google Java Style Guide. Particularly, 3.4.1 is mapped to the Attribute named 'Compliance with the number of top-level class declarations per source file' (1.1.1.4). Note that at the bottom of Fig. 4 the Item *name* and *declaration* are highlighted and linked to the corresponding Attribute *name*. Hence, the 1.1.1.4 *definition* emerges effortlessly from the *name* and *declaration* of the 3.4.1 Item as the "Degree to which the source code file adheres to the number of top-level class declarations per source file that is specified in the coding style guide".

When looking at the NFRsTDO component in Fig. 2, it is important to note that a useful non-taxonomic relationship put into practice in the Define NFRs activity is labeled is mapped to between the terms Statement Item and Attribute. Also, this figure shows that a Statement Item can have a cardinality from none to many Attributes.

Table 7: A set of Google Java Style Guide items mapped to the 'Maintainability' Characteristic, sub-characteristics, and Attributes of a Java source code for Case II

Name of the Guide Sections and Items	**Name of Characteristics, sub-characteristics, and Attributes (*in italics*)**
	1 Maintainability
	1.1 Compliance
3. Source file structure	1.1.1 Source file structure compliance
3.3.3 Ordering and spacing	*1.1.1.1 Compliance with the ordering of types of imports*
	1.1.1.2 Spacing compliance between static and non-static import blocks
	1.1.1.3 Spacing compliance between import sentences
3.4.1 Exactly one top-level class declaration	*1.1.1.4 Compliance with the number of top-level class declarations per source file*
4. Formatting	1.1.2 Formatting compliance
4.1.1 Use of optional braces	*1.1.2.1 Compliance with the use of optional braces*
4.8.2.1 One variable per declaration	*1.1.2.2 Compliance with the number of variables per declaration*
4.3 One statement per line	*1.1.2.3 Compliance with the number of statements per line*
4.4 Column limit: 100	*1.1.2.4 Compliance with the maximum line size*
5. Naming	1.1.3 Naming compliance
5.2.2 Class names	*1.1.3.1 Class naming compliance*
5.2.3 Method names	*1.1.3.2 Method naming compliance*
5.2.6 Parameter names	*1.1.3.3 Parameter naming compliance*

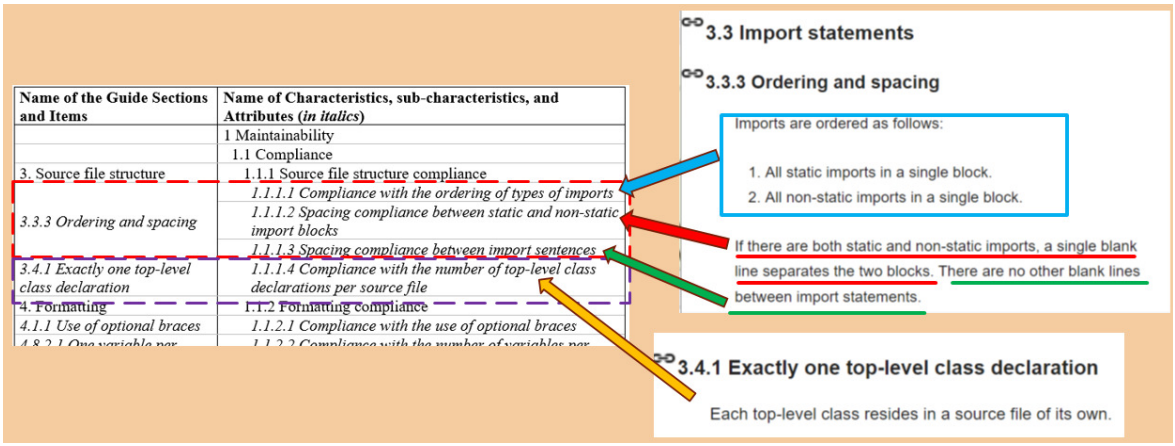

Figure 4: Two highlighted Items from the Google Java Style Guide, with corresponding *name* and *declaration* (put apart on the right), pointing to the four mapped Attribute *names*

From a practical point of view, for each selected Item of the guide, we have defined one or more Attributes. For example, for Item 3.3.3 'Ordering and spacing', we have defined three Attributes as highlighted in Fig. 4. It is important to remark that mapping an Item to more than one Attribute allows the evaluator to perform a more refined measurement and evaluation of the Item. In other words, this correspondence allows more than one Attribute to be associated with a given Item, so that NFRs can be quantified more accurately. Thus, it enables systematic understanding and improvement of source code compliance by using metrics, indicators, and refactoring as methods to Implement Measurement, implement evaluation, and implement change activities, all of which are integrated into an Evaluation Strategy.

In the following paragraphs, instead of showing metrics and indicators for the Attributes related to the Statement Items of Table 7, which are specified in the supplementary material in [27], we will discuss some terms and relationships about NFRs represented in MEvalTDO (Fig. 3) that have a practical impact on the process specification of Evaluation Strategies. In particular, these concepts are common to all Evaluation Strategies included in the family of strategies [28] for different *ME purposes*.

Any Evaluation Strategy can be seen as a resource of an Evaluation Project that includes principles and integrated properties such as an ME Process Specification, an ME Method Specification, and an ME Vocabulary Specification, to help achieve the *ME purpose* of an Evaluation Goal. Examples of *ME purposes* are, namely: to understand, monitor, improve, monitor and control, compare and adopt, and select alternatives, among others. Considering the concept of

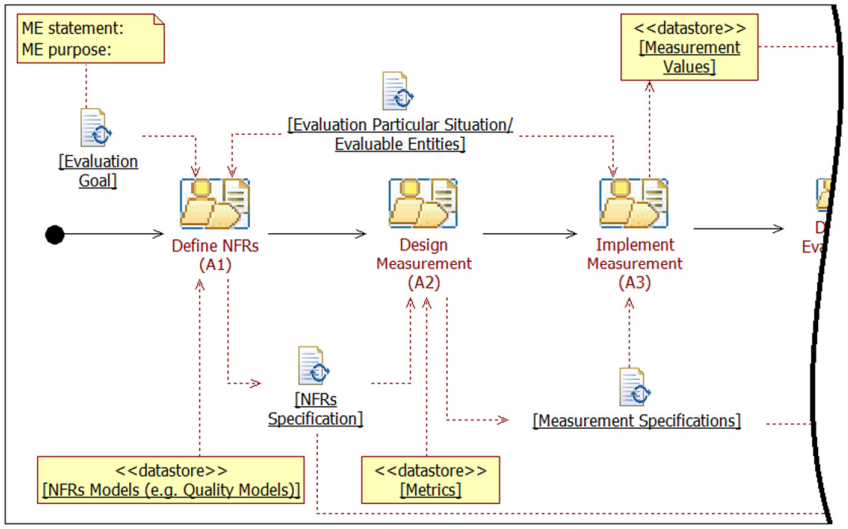

Figure 5: An excerpt from the ME Process Specification that is common to all Evaluation Strategies

ME Process Specification in Fig. 3, Fig. 5 depicts a fragment of a UML activity diagram, which represents a process perspective (view) that includes ME Activities, inputs, outputs, and sequences, among other elements specified in SPEM (*Software & Systems Process Engineering Metamodel*) notation [29].

Looking at Fig. 5, we can see that Define NFRs (A1) is the first specified activity, in which as inputs are: The Evaluation Goal that has an *ME statement* and an *ME purpose*, the current Evaluation Particular Situation with the Evaluable Entities, and the NFRs (Quality) Models –for example in Case II, both the Google Java Style Guide and ISO 25010 quality models are needed to carry out A1. As output, Define NFRs produces the NFRs Specification artifact. In turn, the next activity, Design Measurement (A2), consumes this last artifact and produces the Measurement Specifications artifact, which has Metric specifications as shown in Tables 3 and 4. These Metrics serve as input to Implement Measurement (A3), to obtain Measurement Values as output.

What is worth highlighting from the point of view of practical conceptual impact is that the MEvalTDO common reference vocabulary provides the semantic basis for specifying process views of the work to be performed in an Evaluation Project through the use of an Evaluation Strategy. For example, in Fig. 3, the term Define NFRs with the semantics of Activity is reused in Fig. 5 when labeling A1. Furthermore, looking at Fig. 3, the non-taxonomic relationships of consumes and produces represented for this term are used consistently in Fig. 5, such as the produced artifact of A1 is also called NFRs Specification.

Note that there are aspects of the conceptualization of an ontology that are not fully explicit, so axioms are necessary for its formalization and subsequent implementation of semantic tools. MEvalTDO has 15 axioms specified in first-order logic, which are found in [22]. Below, are 2 of them:

Axiom #1 Description: *A Measurement Value is associated with an ME Non-Functional Requirement (ME_NFR) iff exists an Implement Measurement activity that produces the Measurement Value by consuming an NFRs Specification. Also, the NFRs Specification is produced by the Define NFRs activity by consuming one or more NFRs Models which are part of the ME_NFR.*

$$\forall mv, \exists nfr: MeasurementValue(mv) \wedge ME\_NonFunctionalRequirement(nfr) \wedge isAssociatedWith(mv, nfr) \leftrightarrow \exists im, nfr_s, dnfr, nfr_m: ImplementMeasurement(im) \wedge produces(im, mv) \wedge NFRs\_Specification(nfr_s) \wedge consumes(im, nfr_s) \wedge DefineNFRs(dnfr) \wedge produces(dnfr, nfr_s) \wedge NFRs\_Model(nfr_m) \wedge partOf(nfr_m, nfr)$$

Axiom #2 Description: *If a Measurement Value is associated with an ME Non-Functional Requirement (ME_NFR), then this ME_NFR is an Attribute or a Statement Item, which cannot be both types at the same time.*

$$\forall mv, \exists nfr: MeasurementValue(mv) \wedge ME\_NonFunctionalRequirement(nfr) \wedge isAssociatedWith(mv, nfr) \rightarrow Attribute(nfr) \veebar StatementItem(nfr)$$

Finally, it is worth noting that the Evaluation Strategy used in Cases I and II is called GOCAMEC (*Goal-Oriented Context-Aware Measurement, Evaluation, and Change*) which helps achieve the established improvement purposes. The ME Process Specification in Fig. 5 is a part of the entire GOCAMEC process.

To our knowledge, what is not present in the related works are the relationships of NFRs in the form of Dimensions (sections) and Items of coding style guides with Characteristics and Attributes to build a new Quality Model, all of this supported by ontological concepts, relationships, and axioms.

4.3 Illustrating Case III

The case presented in this subsection illustrates the concepts of Universal Assertions and Universal Things. The last paragraphs of subsection 3.3 describe the concepts of Quality View (a type of NFR View), Quality Focus (a type of Evaluation Focus), and Evaluable Entity Category.

Looking at Fig. 2, we see that the term Quality View (with the semantics of Assertion on Universals) has a Quality Focus (an Assertion as well) and relates to the term Evaluable Entity Category (Universal Thing) through the deals with universals relationship.

Fig. 6 specifies an NFRs View Model, in particular, a Quality View Model with typical Quality Views as well as the depends on and influences relationships between them, useful in SE production lines and Evaluation Projects. This figure indicates that the Software Product Quality View must have the Internal Quality Focus and the Software Product Evaluable Entity Category. For Case I and Case II illustrated above, the Evaluable Entity is the Java Program (with four instances considering before and after improvement) that belongs to the Software Product Evaluable Entity Category. For example, other Evaluable Entities such as an Activity Diagram or an Ontology Conceptualization Document may also fall into this category. In contrast, for example, a compiled entity from 'GUICalculator.java v1.0' belongs to the System Evaluable Entity Category. This category is part of the System Quality View, just like the

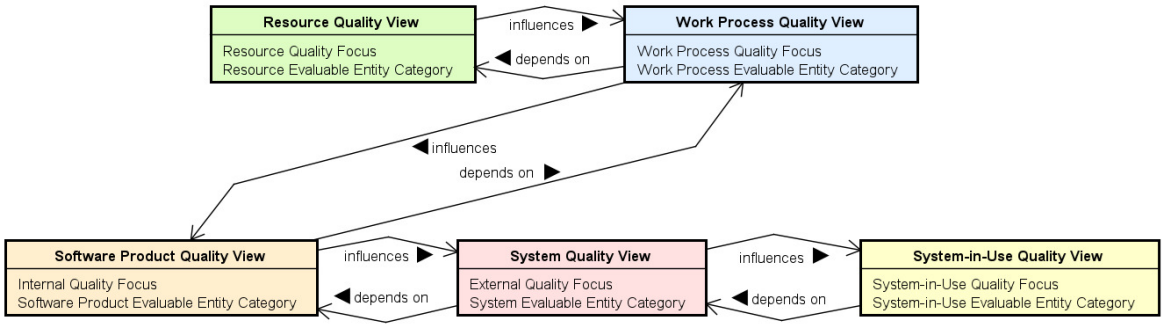

Figure 6: A subset of Quality Views and their relationships specified in a Quality View Model useful in Evaluation Projects

External Quality Focus. Fig. 6 also conveys that the software product quality influences system quality, or vice versa, the latter depends on the former.

Finally, Fig. 2 specifies that a Generic Situation abstracts aspects of Particular Situations. For instance, we can conceive abstractions such as ME strategy patterns that are generic but useful solutions for instantiating specific strategies in evaluation situations/problems considering intentionality.

For Case I and II, the *ME purpose* of 'improve' was established in each Evaluation Goal. It was necessary not only to understand the current situation of the Evaluable Target Entity at hand but also to perform changes on the entity, and then to re-evaluate it in order to gauge the improvement gain. As above mentioned, GOCAMEC was the name of the concrete Evaluation Strategy used as a work resource in both cases regarding only a Quality View and focused on Internal Quality.

Rivera et al. discuss in [13] two Evaluation Strategy patterns called GoMEC_1QV (*Goal-oriented Measurement, Evaluation, and Change for One Quality View*) and GoMEC_2QV (*Goal-oriented Measurement, Evaluation, and Change for Two Quality Views*), which are specifications of Universal Concepts and Assertions. The former is an Evaluation Strategy pattern for improving quality considering just one Quality View, while the latter is devoted to two Quality Views taking into account the influences and depends on relationships.

Fig. 7 specifies the generic process for the GoMEC_2QV strategy pattern where one view is named Dependent Quality View (DQV) and the other Independent Quality View (IQV). As a known customization of GoMEC_2QV,

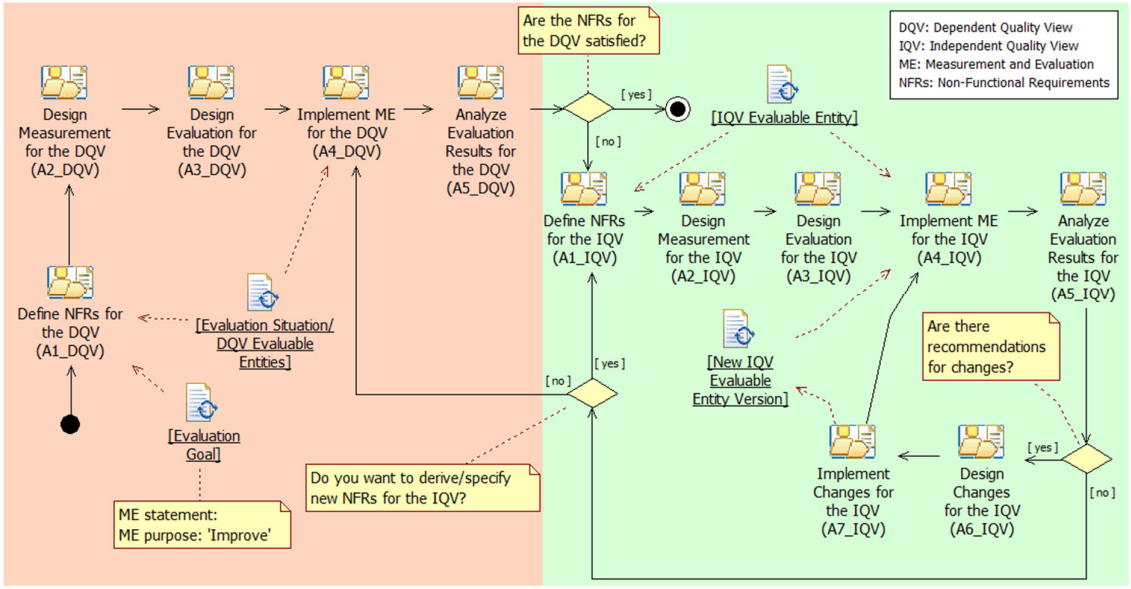

Figure 7: ME Process Specification perspective for the GoMEC_2QV strategy pattern. Note that GoMEC_2QV stands for Goal-oriented Measurement, Evaluation, and Change for Two Quality Views

it was in a software testing company (in Beijing) using SIQinU (*Strategy for Improving Quality in Use*) [30]. This particular Evaluation Strategy (i.e. SIQinU particular activities and methods) only considers the System-in-use Quality View (as DQV) and the System Quality View (as IQV). However, other particular Evaluation Strategies can be conceived from the generic process of Fig. 7, such as the one that considers the Resource Quality View and the Work Process Quality View and their relationships.

More details about Evaluation Strategy patterns devoted to other intentions (goal purposes) for ME, as well as the template with the elements documenting such patterns, can be found in [13].

5 Related Work

In this section, we mainly discuss ontologies for the NFRs domain that simultaneously consider most of the principles or features established in the Introduction Section. These features must be taken into account when designing an ontology as a system of concepts as suggested by D'Aquin et al. [14] and are, namely: i) the use of the conceptual elements of a foundational ontology to specialize and align the developed elements at lower levels; ii) the designed ontology must be modular, extensible, and integrated into a modular ontological architecture or network; iii) the use of graphical representations for the conceptualization promoting also formal simplicity and transparency; and iv) coverage completeness but, at the same time, conciseness and self-intuitiveness of the concepts included.

For example, Rivera et al.'s ontology [13] represents the terms of NFR View, Quality View, Cost View, and NFRs View Model, which were not present in other NFRs ontologies at that time. However, as highlighted at the beginning of this article, it does not meet the first two features mentioned above. The ontology that does meet aspects of the first three features is [5]. Regarding the feature i), the authors align NFRs concepts and relationships with the elements of UFO [11], which is made up of three foundational ontologies, namely: UFO-A (endurants), UFO-B (perdurants or events), and UFO-C (social entities, built on top of UFO-A and B). Taking into account feature ii), this NFRs ontology is in the context of SEON (*Software Engineering Ontology Network*) [31], which locates ontologies organized in layers of generality/specificity. It also complies with feature iii), since it uses a graphical representation based on UML.

The NFRsTDO ontology (Fig. 2) specializes few terms from ThingFO, whereas the NFRs terms presented in [5] specialize/reuse many terms from UFO. While UFO has a large number of terms at the foundational level, in particular around seven terms related to the Quality concept, ThingFO includes three key concepts and related terms but, in particular, only the term Quality-related Assertion, and then meets the conciseness criterion iv). The UFO terms at the foundational level linked to Quality are labeled Quality Universal, Quality Value, Quality Region, Quality Structure, Quality Domain, and Quality Dimension. In our humble opinion, for example, the term Quality Region is not an intuitive foundational term and Quality Value could be delegated at lower levels as observed in MEvalTDO [22] for the terms Measurement Value and Evaluation Value. Therefore, in our ontological approach, we strongly adhere to feature iv) of coverage completeness, but also to conciseness, trying to delegate most responsibilities to the core and in turn top-domain ontologies in order not to overload ThingFO with derivable concepts/terms, relationships, and constraints.

Another related work to quantify NFRs is the MTO ontology, which stands for *Measurement Task Ontology* [32]. The main goal of MTO is to provide a conceptualization for the measurement domain, which is placed in the context of the SEON network and is therefore based on UFO terms. Compared with our proposal, there are several essential terminological differences. To mention just a few, in MTO, on the one hand, the term Measurable Element refers to the aspect to be quantified of a Measurable Entity. In contrast, in NFRsTDO (see Fig. 2), there are terms such as Characteristic, Attribute, and Statement Item as types of the term NFR, which refers to an Evaluable Entity. The rationale and utility of these terms were illustrated in the previous sections.

On the other hand, in MTO, the term Measure is used to refer to the concept of Metric. The same occurs in the vocabulary of the ISO/IEC 25021 standard [33], which replaced the widely used term Metric with Measure. This can lead to confusion, as the term Measure is commonly understood as the value (the output) of a measurement activity. In our approach, we decided to retain the term Metric to refer to the particular form (method) of obtaining a measure or Measurement Value, which aligns with the most widespread use today in academia and industry, as seen in [34] and its subsequent Argentinean ministerial resolution, as well as in [25], [35], and [36], to cite just a few examples.

It is important to highlight that in our proposal, Metric has the semantics of Method, Measurement has the semantics of Activity, Measure has the semantics of Output (which is a Work Product), and Measurement Specification has the semantics of Artifact (which is a Work Product as well). All these terms (Method, Activity, Output, etc.) are more generic terms defined in ProcessCO (see Fig. 1 and [19]), which in turn are each a Particular Thing (a concept defined in ThingFO). In other words, a Metric is a Method, and a Method is a Particular Thing. A Method is a particular way of performing an Activity, which produces a Work Product such as an Output. Therefore, Metric and Measure are related but semantically distinct concepts.

6 Benefits of the Proposal

This section describes the benefits of the present ontological proposal considering the practical impact of ontologies as information and knowledge resources used in the specifications of strategies, processes, methods, techniques, and tools of any engineering discipline such as SE. A non-exhaustive list of benefits is presented below:

• As a concept system, NFRsTDO is represented as an ontology, which implies a richer model structure than a glossary that includes not only terms for concepts but also taxonomic and non-taxonomic relationships and constraints. An ontology supports the human learning and understanding process as a glossary does, but it also provides other benefits, such as structuring and representing knowledge in a more robust, reusable, extensible, and interoperable way, and supporting tools with semantic processing capabilities.

• As a concept system, NFRsTDO is embedded in a multi-level ontological architecture with other interrelated concept systems represented at different levels of generality/specificity, allowing for reuse, extensibility, and alignment of concepts and relationships. In this way, NFRsTDO specializes elements of the core and foundational knowledge basis, allowing an alignment with shared semantic roots, thus fostering cognitive understanding.

• NFRsTDO meets the coverage completeness and conciseness criteria, delegating the responsibility of the concepts for the measurement and evaluation processes, methods, and projects to other ontologies such as the MEvalTDO, MetricsLDO, and IndicatorsLDO ontologies.

• NFRsTDO and related ontologies represent common reference terminologies with practical impact for different specifications. For example, the concepts incorporated in the evaluation strategy process specification shown in Fig. 5 come from terms in the NFRsTDO, MEvalTDO, and MetricsLDO ontologies, while the concepts incorporated in the indirect metric (method) specification template shown in Table 3 also arise from those terms included in the MetricsLDO and NFRsTDO ontologies.

• The multi-level ontological architecture and the included ontologies represent a pyramid of knowledge that sheds light from a minimal set of generic concepts and relationships at the top level to the more specialized and derived terms and relationships at the lower levels.

As a final remark, Gruber [37] says "*For knowledge-based systems, what 'exists' is exactly that which can be represented*". The ontological proposal discussed in [15] and illustrated here for the ontology of NFRs in the context of the multi-level ontological architecture is one way of representing the world as perceived or conceived by human agents, i.e., in this case, by the present authors. But other proposals are also valid in the sense that there is no unique way of representing knowledge of the world taking into account other expert human subjects′ points of view.

7 Conclusions

This paper has mainly analyzed the conceptualization of NFRsTDO, which is a domain-dependent ontology for non-functional requirements concepts located at the top-domain level in the context of a five-tier ontological architecture. This architecture promotes a clear separation of concerns by considering ontological levels that allow for proper mapping of conceptual components (ontologies) and encourages modularity and consistent reuse and extension of ontological elements (primarily terms and relationships) at all lower levels.

Since NFRsTDO is located at the top-domain level, we have highlighted that the components MEvalTDO and TestTDO depend on some terms in NFRsTDO, and there is also a double link between FRsTDO and NFRsTDO. Besides, at the low-domain level, MetricsLDO and IndicatorsLDO reuse some of NFRsTDO terms. Thus, MetricsLDO and IndicatorsLDO include terms that support ME processes and methods for producing data and information values from NFRs specifications –i.e., from Attributes, Statement Items, and Characteristics– that are then used for analysis and recommendations.

Another issue we would like to point out is that the term NFR does not represent a concrete entity (Particular Thing), but rather a Quality-, Constraint-related Assertion of a certain aspect of the entity to be measured and evaluated. Also, an NFR as an Assertion can deal with a category of concrete entities (Universal Thing) as a categorical abstraction.

Furthermore, we have discussed not only the key concepts and relationships for the NFRs ontology but also their applicability to particular and universal entities. To this end, we have illustrated three typical cases in different particular situations in measurement and evaluation projects carried out both in the academic and industrial context.

In future work, we will discuss and illustrate the practical impact of the concepts and relationships included in the NFRsTDO component that the TestTDO ontology takes into account. The reader can surmise that the term Test Requirement has the semantics of both Functional Requirement (FRsTDO term) and Non-Functional Requirement (NFRsTDO term) in order to specify functional and nonfunctional testing approaches.

Acknowledgments

This paper is a substantial extension made of that published in [18] considering the kind invitation of CLEI'24 chairs to resubmit an extended manuscript. This work is partially supported by project 09/F079 of the Engineering School at UNLPam, and also project POIRe 2023-9 at UNLPam.

References

[1] ISO 704, *Terminology work – Principles and methods*, 2022.

[2] G.-C. Roman, "A taxonomy of current issues in requirements engineering", In: *IEEE Computer*, vol. 18, no. 4, pp. 14–23, April 1985, Available: https://doi.org/10.1109/MC.1985.1662861.

[3] R. Veleda and L.M. Cysneiros, "Towards an Ontology-Based Approach for Eliciting Possible Solutions to Non-Functional Requirements", In: Giorgini, P., Weber, B. (eds) *Advanced Information Systems Engineering. CAiSE 2019. Lecture Notes in Computer Science*, vol 11483. Springer, Cham., 2019. Available: https://doi.org/10.1007/978-3-030-21290-2_10.

[4] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, *Non-functional Requirements in Software Engineering*, Kluwer Academic Publishers, 2000.

[5] R. Guizzardi, F.-L. Li, A. Borgida, G. Guizzardi, J. Horkoff, and J. Mylopoulos, "An Ontological Interpretation of Non-Functional Requirements", *Frontiers in Artificial Intelligence and Applications*, E-book: Formal Ontology in Information Systems, Vol. 267: pp. 344–357, 2014. Available: https://doi.org/10.3233/978-1-61499-438-1-344.

[6] J. C. S. do Prado Leite and A.P. Franco, "A Strategy for Conceptual Model Acquisition", *Proceedings of the 1st IEEE Int. Symp. on Requirements Engineering* (RE'92), IEEE Computer Society Press, pp. 243–246, 1993. Available: https://doi.org/10.1109/ISRE.1993.324851.

[7] L. M. Cysneiros and J. C. S. do Prado Leite, "Nonfunctional requirements: from elicitation to conceptual models", *IEEE Transactions on Software Engineering*, 30:(5), pp. 328–350, 2004. Available: https://doi.org/10.1109/TSE.2004.10.

[8] M. Jackson and P. Zave, "Deriving specifications from requirements: an example", *17th International Conference on Software Engineering, ICSE 1995*, Seattle, Washington, ACM, New York, pp. 15–24, 1995.

[9] I. J. Jureta, J. Mylopoulos, and S. Faulkner, "A core ontology for requirements", *Applied Ontology*, 4:(3), pp. 169–244, 2009.

[10] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari, "Ontology Library", WonderWeb Deliv. D18, 2003.

[11] G. Guizzardi, "Ontological foundations for structural conceptual models", PhD thesis, University of Twente, Enschede, The Netherlands, ISBN 90-75176-81-3, 2005.

[12] L. Olsina, M.F. Papa, and H. Molina, "Ontological Support for a Measurement and Evaluation Framework", *International Journal of Intelligent Systems*, Wiley-Blackwell (R. Yager Ed.) 2:(12), pp. 1282–1300, 2008, Available: https://doi.org/10.1002/int.20320.

[13] M.B. Rivera, P. Becker, and L. Olsina, "Quality Views and Strategy Patterns for Evaluating and Improving Quality: Usability and User Experience Case Studies", *Journal of Web Engineering*, Rinton Press, US, 15:(5&6), pp. 433–464, ISSN 1540-9589, 2016.

[14] M. D'Aquin and A. Gangemi, "Is there beauty in ontologies?", *Applied Ontology*, 6:(3), pp. 165–175, 2011.

[15] L. Olsina, "The Foundational Ontology ThingFO: Architectural Aspects, Concepts, and Applicability", In: Fred, A., Aveiro, D., Dietz, J., Bernardino, J., Masciari, E., Filipe, J. (eds.) *Knowledge Discovery, Knowledge Engineering and Knowledge Management. IC3K 2021. Communications in Computer and Information Science*, vol 1718, Springer. Chapter 5, pp. 73–99, 2023.

[16] ISO/IEC 25010, *Systems and Software Engineering – Systems and software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models*, 2011.

[17] Google Java Style Guide. Available: https://google.github.io/styleguide/javaguide.html, and last accessed by

November (2024).

[18] L. Olsina, P. Becker, M.F. Papa, and Rossi, G, "Concepts and Applicability of Non-Functional Requirements for Particular and Universal Things", *50th Latin American Computer Conference (CLEI'24)*, Bahia Blanca, Argentina, IEEE Xplore, pp. 1–10, 2024. Available: https://doi.org/10.1109/CLEI64178.2024.10700118.

[19] P. Becker, M.F. Papa, G. Tebes, and L. Olsina, "Discussing the Applicability of a Process Core Ontology and Aspects of its Internal Quality", *Software Quality Journal*, Springer, 30:(4), pp. 1003–1038, 2022. Available: https://doi.org/10.1007/s11219-022-09592-3.

[20] S. Fleetwood, "The ontology of things, properties and powers". In: *Journal of Critical Realism*, 8:(3), pp. 343–366, 2009.

[21] L. Olsina, M.F. Papa, and P. Becker, "NFRsTDO v1.2's Terms, Properties, and Relationships -- A Top-Domain Non-Functional Requirements Ontology", pp. 1–9, 2023. Available: https://doi.org/10.48550/arXiv.2302.01096.

[22] L. Olsina, M.F. Papa, and P. Becker, "MEvalTDO 2.0's Concepts, Relationships, and Constraints -- A Measurement and Evaluation Top-Domain Ontology", pp. 1–26, 2024. Available: https://doi.org/10.13140/RG.2.2.30184.07689.

[23] P. Becker, L. Olsina, and M.F. Papa, "Especificación de Métricas de Mantenibilidad para Mejorar Código Java: Caso Aplicado en un Curso Avanzado de Ingeniería en Sistemas", *Actas del XXI Congreso Nacional de Ingeniería Informática / Sistemas de Información (CoNaIISI)*, Tucumán, Argentina, pp. 562–575, ISBN: 978-987-8992-38-9, 2023. Available: https://bit.ly/RG-Metricas_de_Mantenibilidad.

[24] G. Covella and L. Olsina, "*Assessing Quality in Use in a Consistent Way*", In: *ACM proceedings of the Int'l Congress on Web Engineering, (ICWE'06)*, San Francisco, USA, pp. 1–8, 2006. Available: https://doi.org/10.1145/1145581.1145583.

[25] G. A. Campbell, "Cognitive Complexity — An Overview and Evaluation", *IEEE/ACM International Conference on Technical Debt (TechDebt)*, Gothenburg, Sweden, pp. 57–58, 2018.

[26] T. J. McCabe, "A complexity measure", *IEEE Transactions on Software Engineering*, Vol. SE-2, No. 4, pp. 308–320, 1976.

[27] P. Becker, M.F. Papa, and L. Olsina, "Approach to Improving Java Source Code considering Non-compliance with a Java Style Guide", Springer book: *Computer Science – CACIC 2022, Communications in Computer and Information Science, CCIS 1778*, Chapter 9, pp. 123–139, 2023.

[28] L. Olsina and P. Becker, "Family of Strategies for different Evaluation Purposes", In: *20th Ibero-American Conference on Software Engineering* (CIbSE'17) held in the framework of ICSE, CABA, Argentina, Published by Curran Associates 2017, pp. 221–234, 2017.

[29] OMG-SPEM, *Software & Systems Process Engineering Meta-Model Specification v2.0*, 2008. Available: https://www.omg.org/spec/SPEM/2.0/PDF.

[30] P. Lew, L. Olsina, P. Becker, and L. Zhang, "An Integrated Strategy to Systematically Understand and Manage Quality in Use for Web Applications", *Requirements Engineering Journal*, Springer London, 17:(4), pp. 299–330, 2011.

[31] F.B. Ruy, R.A. Falbo, M.P. Barcellos, S.D. Costa, and G. Guizzardi, "SEON: A software engineering ontology network", *20th International Conference on Knowledge Engineering and Knowledge Management*, pp. 527–542, 2016.

[32] L.A. Santos, M.P. Barcellos, R. A. Falbo, C. C. Reginato, and P. C. Campos, "Measurement Task Ontology", *OntoBras'19*, Published by CEUR-WS. Vol. 2519, Paper 8, 2019.

[33] ISO/IEC 25021, *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality measure elements*, 2012.

[34] CONFEDI, Propuesta de estándares de segunda generación para la acreditación de carreras de ingeniería en la República Argentina, *Libro rojo de CONFEDI*, Rosario, Argentina, 2018. Available: https://confedi.org.ar/download/documentos_confedi/LIBRO-ROJO-DE-CONFEDI-Estandares-de-Segunda-Generacion-para-Ingenieria-2018-VFPublicada.pdf.

[35] V. Lenarduzzia, T. Kilamob, and A. Janes, “Does Cyclomatic or Cognitive Complexity Better Represents Code Understandability? An Empirical Investigation on the Developers Perception”, 2023. Available: https://doi.org/10.48550/arXiv.2303.07722.

[36] M. Muñoz Baron, M. Wyrich, and S. Wagner, “An empirical validation of cognitive complexity as a measure of source code understandability”, In: *14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pp. 1–12, 2020.

[37] T. R. Gruber, “A Translation Approach to Portable Ontologies”, *Knowledge Acquisition, 5*:*(2),* pp. 199–220, 1993.